Context-aware Provision of Advanced Internet Services

Raffaella Grieco Delfina Malandrino
Università degli Studi di Salerno, Italy
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"

Francesca Mazzoni
Università degli Studi di Modena e Reggio Emilia, Italy
Dipartimento di Ingegneria dell'Informazione

Daniele Riboni
Università degli Studi di Milano, Italy
Dipartimento di Informatica e Comunicazione

Abstract

The pervasive and mobile computing scenario is characterized by the high heterogeneity of devices used to access services, and by frequent changes in the user's context. In this paper we present an architecture, realized on top of two existing frameworks, whose main goal is to support the provisioning of advanced, context-aware Internet services in such a scenario. In order to demonstrate the feasibility of our solution, we have developed a prototype location-based service addressed to mobile users that takes advantage of this architecture.

1 Introduction

In the *Pervasive and Ubiquitous Computing* era, the trend is to provide Web content and multimedia applications by taking into account four important requirements: *anytime-anywhere* access to *any data* through *any device* and by using *any access network*. As a consequence, it is increasingly significant to provide tailored content efficiently, in order to address the mismatch between the rich multimedia content that is widely available and the limited client capabilities and available bandwidth. Furthermore, users feel the need of personalized content that matches their personal preferences, considering not only device capabilities and network status, but a wider notion of *context*, which includes -among other things- their current location, activity, and surrounding environment.

One of the current research topics in distributed systems is how to extend the traditional client/server computational paradigm in order to allow the provision of *intelligent* and *advanced* services. To this aim, new actors are introduced into the WWW scene, in the form of intermediaries that act on the HTTP data flow exchanged between client and server. Several examples of intermediary adaptation systems exist in the literature, such as iMobile by AT&T [7], whose main goal is to provide personalized mobile services; the TACC infrastructure [4], whose main goal is to support the dynamic deployment of transformation modules into proxy-based components; RabbIT[1] and Privoxy[2], whose main goal is to provide functionalities such as text and image transcoding, removing of cookies, GIF animations, advertisement, Java Applets, etc. Many of these systems provide adaptation functionalities without taking into account user's preferences and context information.

In order to support the provisioning of this type of services, we have defined and experimented with an architecture that builds on top of two existing frameworks: the *Scalable Intermediary Software Infrastructure (SISI)* [3], and the *Context Aggregation and REasoning (CARE)* [1] middleware. Our proposal is supported by a running implementation, and by a prototype service aimed at providing location-based support to mobile users.

The rest of the paper is organized as follows. Section 2 illustrates the architecture and the data flow; Section 3 describes the dynamic adaptation mechanism; Section 4 describes the prototype service.

2 Architecture overview

In this section we present the architecture for context-aware service provisioning. The main components of the

[1] http://rabbit-proxy.sourceforge.net
[2] http://www.privoxy.org

architecture are the *SISI* intermediary infrastructure and the *CARE* middleware for context awareness.

2.1 Scalable Intermediary Software Infrastructure (SISI) overview

The main goal of the SISI project [3, 5] is to create a new framework that aims at facilitating the deployment of efficient and programmable adaptation services running on edge servers on the WWW.

This framework, built on top of Apache Web server and mod_perl, provides a modular architecture that allows an easy definition of new functionalities implemented as building blocks in Perl. These building blocks, packaged into *Plugins*, produce transformations on the information stream as it flows through them. Moreover, they can be combined in order to provide complex functionalities. Thus, multiple Plugins can be composed into SISI edge services, and their composition is based on preferences specified by end users. The SISI architecture consists of five main modules, which are briefly described in the following.

The *ProxyPerl Module* intercepts all clients requests and initializes the transaction process. Its most important task is to fetch the requested URLs manipulating, if necessary, HTTP headers. If no transformation has to be applied on the HTTP flow, the requested document will be sent back to the client. Otherwise, the transaction will be managed by the handlers invoked according to the user's context data, obtained from CARE (as explained in Section 2.2).

The *Authorization Module* is useful both to restrict access to the proxy server as well as to distinguish between users, so that each user can have different SISI services applied to her requests.

The *Apache Registry Handler* is the standard Apache module that allows to efficiently execute CGI scripts under mod_perl; it is used by SISI to handle users' and services data.

The *FilterPlugin Module* acts as a dispatcher within our architecture by activating the services according to the users' preferences.

Finally, the *Deploy Module* is used by the programmer to add new services to the framework. It consists of an automatic modules generation process that implements intermediary services starting from simple Perl files.

We already implemented many adaptation services, which are extensively described in [5]; for the sake of this paper, we cite the FilterImg service, which retrieves the origin image from the Web and performs some adjustments, such as changing colors and contrast, rotating, cropping and/or resizing.

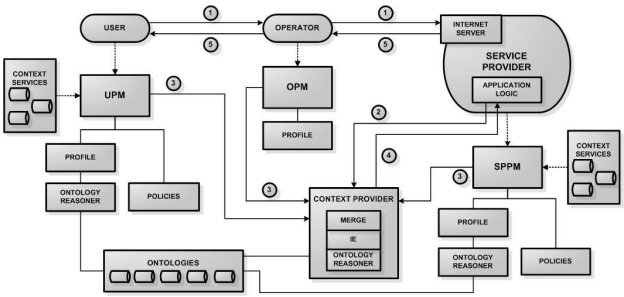

Figure 1. The CARE middleware.

2.2 The CARE middleware for context-awareness

The CARE middleware aims at supporting context-aware mobile services in an environment characterized by distributed context sources. CARE has been designed in order to support a wide range of technologies and application servers. In the described architecture, CARE is in charge of providing SISI with those context data that are useful to determine the activation and the parameters of services.

Figure 1 shows an overview of the CARE middleware, which is presented in more detail in [1]. We call *profile* a subset of context information collected and managed by a certain entity. Profiles are represented by using the CC/PP language [6], and are managed by three entities, namely: the user with her devices; the network operator with its infrastructure; and the service provider. Each entity has a dedicated profile manager for handling its own context data: the User Profile Manager (UPM); the Operator Profile Manager (OPM); and the Service Provider Profile Manager (SPPM). The middleware comprises ontology services for managing and reasoning with socio-cultural context data that cannot be modeled in CC/PP. Adaptation and personalization parameters are determined, at the time of the service request, by policy rules defined by both the user and the service provider, and managed by their corresponding profile managers. The CONTEXT PROVIDER module is in charge of calculating the aggregated context information that will be used by the application logic for adapting the service. In particular, it retrieves context data from the profile managers, and evaluates adaptation policies solving possible conflicts arising among context data and/or policies. The ad-hoc rule-based reasoning services of the CONTEXT PROVIDER are particularly efficient. Experimental results [2] have shown that the evaluation of policy rules is executed in few milliseconds.

2.3 Data flow

Figure 2 shows the data flow upon a user request. In order to provide the CONTEXT PROVIDER with the user's

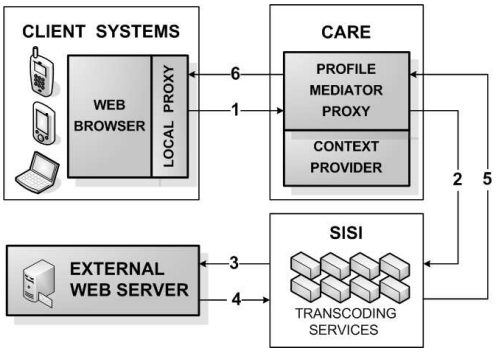

Figure 2. The adaptation architecture.

identification and with the information required to retrieve distributed profiles and policies, the request is intercepted by a LOCAL PROXY that adds these data into custom HTTP headers. Then (Step 1), the request is forwarded to the PROFILE MEDIATOR PROXY of CARE, which retrieves the aggregated context data from the CONTEXT PROVIDER module. These data -which include the list of SISI services to be applied, as well as their parameters- are inserted into the HTTP headers, and the request is sent to the SISI architecture (Step 2). SISI retrieves the requested resource from the external Web server (Steps 3 and 4), and applies the services on the basis the user's context data. Finally, the adapted resource is sent back to the user's client (Steps 5 and 6).

3 Service activation policies

By default, each SISI service is deactivated. The activation of services, as well as the service parameters, are determined by policy rules declared by both the user and the service provider. Policies are essentially conditions on context data that set the value of other profile attributes when satisfied.

Example 1 *Consider the case of the* FilterImg *service for image transcoding. The evaluation of the following policy rules determine the activation state and the parameters of the service:*

R1: ***If*** DeviceType = *CellPhone*
Then Set FilterImg:Activate=*'on'*

R2: ***If*** DeviceType = *CellPhone* ***And*** Bearer ! = *UMTS*
Then Set FilterImg:removeImage=*'on'*

R3: ***If*** ColorCapable = *no*
Then Set FilterImg:black&white=*'on'*

Rule R1 is used to activate the service for devices with low capabilities (cell phones). Rule R2 instructs SISI to remove images when the cell phone is not connected through a UMTS bearer. Rule R3 is used to determine the transformation in black and white of images for devices that cannot display colors. It should be noted that the user's preferences regarding the activation of services, as well as their parameters, are represented through a proper CC/PP vocabulary.

Since policies can be declared by both the user and the service provider, conflicts can arise between policies setting a value for the same service parameter. When conflicting policies are declared by the same entity, the conflict is easily solved by allowing the entity to specify an explicit priority between conflicting rules. On the other hand, when a conflict is encountered between a policy declared by the user and a policy declared by the service provider, this type of conflict is resolved on the basis of *resolution directives* declared by the service provider (see [2] for more details). With regard to the SISI services, our choice was to give the highest priority to the user, while letting the service provider to specify default policies.

Example 2 *Consider the case of the* FilterImg *service, and suppose that the rules of Example 1 are the default rules declared by the service provider. Suppose that a particular user declares the following policy rule to disable the removal of images when the cell phone is connected through GPRS:*

R4: ***If*** DeviceType = *CellPhone* ***And*** Bearer = *GPRS*
Then Set FilterImg:removeImage=*'off'*

When a cell phone is connected through GPRS, both rules R2 and R4 could fire, determining conflicting values (i.e., 'on' and 'off') for the same parameter. In this case, since our choice was to give higher priority to the user's policies, rule R4 is evaluated first, and sets the value 'off' to the FilterImg:removeImage *parameter. Since rule R4 fired, rule R2 is discarded.*

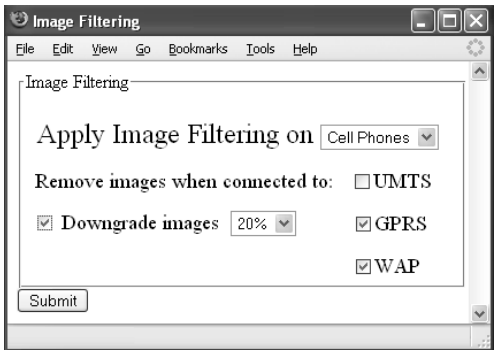

Figure 3. User interface to declare policies.

Of course, policies expressed by means of the formal language shown in Examples 1 and 2 are not intuitive for the final users of the system. For this reason, users are provided with Web interfaces to manage their preferences regarding the activation and parameters of SISI services. Figure 3 shows the interface to express preferences regarding the *FilterImg* service.

4 A prototype service

In order to show the integration between CARE and SISI we have implemented a prototype location-based service, named *GeoAware*. This service is addressed to mobile users equipped with a mobile device and a GPS receiver. The main goal of this service is to provide a map with information about both the current location of the user and the locations (expressed as physical addresses) appearing on the Web page she is currently viewing. The main steps performed by the GeoAware service are described below.

At first, CARE retrieves the GPS coordinates of the user from her profile manager, and communicates them to SISI as explained in Section 2.3.

The GeoAware service parses the requested Web page and, by applying regular expressions, matches all standard U.S. addresses [8]. When an address is recognized, GeoAware adds a hyperlink on the Web page, highlighted by a particular icon (see Figure 4-A). When the user selects the hyperlink icon, GeoAware invokes the *geocoder.us* service[3] to obtain the coordinates of the relative address. Geocoder.us is a public service providing free geocoding of addresses in the United States, and relies on Geo::Coder::US, an open-source Perl module available from CPAN.

After having obtained this information, GeoAware builds a query string to be issued to the U.S. Census Bureau *TIGER Map Server*[4], which provides public-domain, customized U.S. maps. Figure 4-B shows the map obtained from the address in Figure 4-A. The current position of the user is represented by a blue star with a label YOU ARE HERE, while the position corresponding to the address is represented by a red star labeled with the address. Exploiting the other SISI services, the map is properly adapted considering device capabilities and available bandwidth.

The interest for geolocalized services is witnessed by the recent service provided by Google with its AutoLink button in the Google toolbar. Even if that service addresses the same needs as ours, it is currently available only to users with desktop browsers. Moreover, the AutoLink functionalities are somewhat limited since it does not take advantage of a context-awareness framework.

[3] `http://geocoder.us/`

[4] `http://tiger.census.gov/cgi-bin/mapbrowse-tbl`

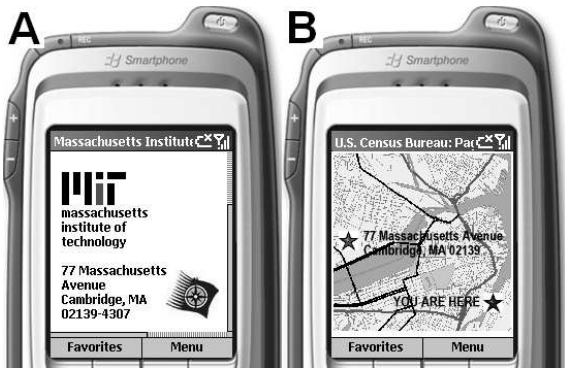

Figure 4. The GeoAware service.

References

[1] A. Agostini, C. Bettini, N. Cesa-Bianchi, D. Maggiorini, D. Riboni, M. Ruberl, C. Sala, and D. Vitali. Towards Highly Adaptive Services for Mobile Computing. In *Proc. of IFIP TC8 Working Conference on Mobile Information Systems (MOBIS)*, pages 121–134. Springer, 2004.

[2] C. Bettini and D. Riboni. Profile Aggregation and Policy Evaluation for Adaptive Internet Services. In *Proc. of The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous)*, pages 290–298. IEEE Computer Society, 2004.

[3] M. Colajanni, R. Grieco, D. Malandrino, F. Mazzoni, and V. Scarano. A Scalable Framework for the Support of Advanced Edge Services. In *Proc. of the 2005 Int. Conf. on High Performance Computing and Communications (HPCC'05)*, pages 1033–1042, Sorrento (NA), Italy, September 2005.

[4] A. Fox, Y. Chawathe, and E. A. Brewer. Adapting to Network and Client variation using active proxies: Lessons and perspectives. *IEEE Personal Communications*, 5(4):10–19, 1998.

[5] R. Grieco, D. Malandrino, F. Mazzoni, and V. Scarano. Mobile Web Services via Programmable Proxies. In *Proc. of the IFIP TC8 Working Conference on Mobile Information Systems - 2005 (MOBIS), To appear*, Leeds (UK), December 2005.

[6] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation, W3C, January 2004.

[7] C. Rao, Y. Chen, D.-F. Chang, and M.-F. Chen. imobile: A proxy-based platform for mobile services. In *Proceedings of the First ACM Workshop on Wireless Mobile Internet (WMI 2001)*. ACM Press, 2001.

[8] US Postal Service. Postal Addressing Standards. Technical Report Publication 28, November 2000.